Universidade Federal da Paraíba
Centro de Ciências Aplicadas e Educação
Departamento de Ciências Exatas
Bacharelado em Sistemas de Informação
Licenciatura em Ciências da Computação

Relatório do projeto Banco Imobiliário (Etapa 2 - Stories 5, 6 e 7)

Disciplina: Programação Orientada a Objetos
Equipe: Equipe 2
Link Github: dcx-cursos/projeto-poo-2019-1-jo/tree/dev
Membros: Amanda Azevedo Martins
Clebson Augusto Fonseca
Joana Darck Soares da Silva
Joyce Sousa dos Santos

1. Introdução

1.1. Objetivos

O seguinte relatório apresenta projeto de Banco Imobiliário, que foi definido, pelo professor Fábio Morais, como terceira nota da disciplina de Programação Orientada a Objetos. Assim como, o relato do processo de desenvolvimento, as funcionalidades, as tecnologias usadas, uma breve descrição sobre do modelo do sistema (design) e como as classes e as interfaces se relacionam, da implementação que foi efetuada para a segunda entrega deste projeto.

1.2. Tecnologia

As tecnologias usadas para a conclusão desta segunda entrega foi, a linguagem de programação Java8, a IDE (*Integrated Development Environment*) "Eclipse Java 2019-06" para desenvolvimento do projeto. Utilizamos o GitHub, para armazenamento e versionamento do código. Além disso utilizamos a *framework* de testes, *JUnit5,* o *Mockito* com a finalidade simulação de algumas classes, para que podermos testar algumas funcionalidades e para criação do diagrama de classes em UML (*Unified Modeling Language*), utilizamos o "Astah UML".

1.3. Entregas e Datas

As datas de entrega deste projeto está seguindo o calendário estabelecido pelo professor Fábio, sendo assim, conseguimos entregar a primeira versão dia 12 de agosto de 2019 e a segunda entrega foi feita dia 02 de setembro de 2019.

2. Metodologia

Primeiramente, foi desenvolvido os testes da primeira entrega, que ficaram faltando, e em seguida implementamos os Stories 5, Sorte ou Revés, 6, Prisão e 7, Companhias. Simultâneo a isso foram desenvolvidos os testes do código que estava sendo implementado e os mocks. Posteriormente documentamos o código, aprimoramos a segunda versão da UML e redigimos o relatório.

3. Desenvolvimento

3.1. Funcionalidades disponíveis

Na primeiro entrega o Banco Imobiliário possuía as funcionalidades que foram descritas nas Users Stories de número 1 a 4,

Criar um novo jogo, Jogada no Banco Imobiliário, Status e Compra de títulos de propriedades. Sendo assim, o banco imobiliário descrito neste relatório possuía as funcionalidades de cadastrar novos jogadores (dentro do intervalo de 2 à 8 jogadores), iniciar o jogo, jogar dados, avançar os peões dos jogadores no tabuleiro, sair do jogo, ver o *status* do jogador no jogo, e por fim possibilitar a compra títulos de propriedades do banco e ao parar em uma propriedade que já possui dono, ter que pagar um aluguel.

Com esta segunda entrega adicionamos as funcionalidades descritas nas Users Stories de número 5 a 7, Sorte ou revés, Prisão e Companhias.

Com isso o jogador pode comprar títulos de companhias e pagar o aluguel por ter parado em uma companhia que já possuía um dono.

Ao parar em uma posição do tabuleiro de Sorte ou Revés, é possível puxar uma carta da pilha de Sorte ou Revés e receber, ou ter que pagar, dinheiro ao banco, a depender se a soma dos dados é um número ímpar ou par. Como também podemos apenas receber do banco ou de todos os outros jogadores, pagar dinheiro ao banco, poder ficar livre da prisão, ou ir para a prisão, e até mesmo ir para a primeira posição do tabuleiro.

A última funcionalidade implementada nesta entrega foi a da prisão, ao parar nela o jogador fica impossibilitado de avançar no jogo, a menos que ele possua uma carta de sorte ou revés que o livre da prisão, ele pague a fiança ou consiga tirar números iguais ao jogar os dados.

3.2. Padrões de projeto implementados

3.2.1. Padrão Facade

3.2.1.1. Jogo Facade

Esta classe é a fachada, a qual usamos para manipular o jogo. Tem o propósito de promover uma interface unificada e ajuda a reduzir a complexidade do sistema.

3.2.2. Padrão Factory

3.2.2.1. Jogo Factory

Nesta classe tem o método onde analisa as opções que o jogador tem. Em tempo de execução o programa vai executar a classe certa.

3.2.2.2. Jogo Factory Prisão

Acontece o mesmo que na classe “JogoFactory”, a diferença é que só é utilizada quando o jogador cai na prisão.

3.2.2.3. Título Factory

Antes utilizamos uma interface “Titulo”, porém agora estamos usando o factory method para evitar códigos duplicados, assim, reaproveitando os códigos. TituloFactory é uma classe abstrata com métodos que os títulos têm em comum

3.3. Descrição do design do sistema

3.3.1. Pacote ufpb.jogo

Este pacote é o pacote que possui os principais classes : Conta - onde vai controlar o saldo de cada jogador; Dado, para jogar os dados e definir quantas casas no tabuleiro o jogador vai avançar; Jogador, onde vai definir as características de cada jogador(nome, cor, etc.); Jogo - mostra o que acontece durante a partida; JogoFacade é a fachada, onde usamos para manipular o jogo; JogoFactory é onde tem o método que analisa as opções; JogoFactoryPrisao, se o jogador estiver na prisão, então será usada essa classe para analisar as opções disponíveis; o Main;, onde vai iniciar o jogo; e Tabuleiro, onde o jogador vai se movimentar ao decorrer da partida.

3.3.2. Pacote ufpb.opcoes

Este pacote é referente às opções que cada jogador tem durante sua jogada. Sendo elas, a opção de jogar - jogar os dados e se movimentar no tabuleiro, de ver o status do jogador(se ele tem títulos, mostrar quais são; o saldo do jogador e a posição no qual está situado), de tentar usar a carta de Habeas Corpus para sair da prisão, de pagar para sair da prisão, de jogar os dados para tentar sair da prisão e a opção de sair do jogo.

3.3.3. Pacote ufpb.cartas

Primeiramente, como há muitas cartas de sorte ou revés que realizam diversas ações, agrupamos as cartas em grupo, de acordo com o tipo de ação que ela realiza. Conseguimos agrupar as cartas como sendo do tipo Pague, em que o jogador tem que pagar algum valor ao banco, Presente, em que todos os jogadores pagam alguma quantia

ao jogador que retirou esse carta da pilha, Receba, em que o jogador recebe uma quantia do banco, Vá para prisão, em que o jogador é mandado para a prisão, Habeas Corpus, em que o jogador ao possuir esta carta pode-se livrar da prisão e Sorte ou Revés, que a depender do resultado da soma dos dados, ou ele paga ou recebe alguma quantia do banco.

Neste pacote são armazenadas as classes que extends a superclasse, Sorte ou revés. Nela foi criada uma classe para cada tipo descrito acima, que realizam ações com base na sua descrição.

3.3.4. Pacote ufpb.lougradouros

De maneira semelhante às cartas, foi feito uma classe para cada tipo de posição do tabuleiro, os tipos definidos foram, companhia, imposto de renda, lucros ou dividendos, parada livre, ponto de partida, posição de sorte ou revés, prisão e terreno.

Como todas os tipos definidos são posições do tabuleiro, então todos os tipos herdam de uma interface, que foi intitulada de posição, essa interface é o que possibilita que aconteçam os eventos decorridos da parada do jogador em uma nova posição do tabuleiro. Além disso, há no pacote a interface título, que a principal função é a compra de títulos de propriedades e companhias do banco.

3.3.5. Pacote ufpb.exception

Este pacote é referente ao armazenamento todas essas classes do tipo Exception existentes no sistema de banco imobiliário, tais como, se a cor que um jogador escolheu para si é uma cor possível, se um valor de dinheiro é válido , se algum limite foi excedido, entre outros. Este pacote possibilita que existam exceções mais especializadas, para que possamos tratá las para entender melhor alguns erros.

3.3.6. Pacote ufpb.recuperaDados

Este pacote é referente a camada de persistência de dados em arquivos, onde conseguimos recuperar os dados, em forma de String, de arquivos *.TXT* para serem

convertidos em objetos em outras classes. Este pacote, possibilita que haja uma separação entre a camada de *Business Logic* e a de *Data*, deixando assim o código mais limpo.

4. Diagrama de classes

4.1. Diagrama de classes da segunda entrega do jogo

5. Testes

Utilizamos o *framework* de testes, *JUnit5* juntamente com o *Mockito* com a finalidade de simulação de algumas classes, assim testando algumas funcionalidades. Até o momento está implementado os testes das classes Jogador, Conta e JogoFacade.

Na classe ContaTest, está testando o saldo após ser depositado ou debitado um certo valor. Na JogadorTest, vai testar quando jogador lança os dados e vai para a posição certa. A classe JogoFacadeTest tem testes para remover e adicionar um jogador verificar se o jogador está na prisão e para verificar se a cor escolhida é válida.